A Hybrid DeBERTa and Rule-Based Pipeline for Personalized Academic Highlight Detection

William Johnston and Claude (Anthropic)

Abstract

We present a two-stage hybrid pipeline for automatically detecting and classifying highlight-worthy spans in academic text, trained to replicate the highlighting style of a specific reader. The system combines a fine-tuned DeBERTa-v3-large token classifier with rule-based keyword and citation pattern matching, followed by a Gradient Boosting second-stage filter trained on 80,000 labeled spans to match the user's selectivity. On a held-out evaluation of 1,290 ground truth spans across 11 annotated documents, the pipeline achieves 89.8% recall at a 4.9x prediction-to-ground-truth ratio, with 100% recall on individual test files. The system supports eight highlight types (normal, important, really important, reference, confusing, pretty good, pretty bad, funny), processes text at approximately 35 lines per second on a single GPU, and includes end-to-end tooling for PDF-to-JSONL conversion and PDF annotation rendering. The complete pipeline has been validated on a corpus of 6,779 academic papers and 341 books.

1. Introduction

Academic reading involves selective attention: readers highlight passages that are surprising, important, or worth revisiting. This highlighting behavior is highly personal, reflecting individual knowledge, interests, and reading goals. Automating this process requires not only detecting objectively noteworthy content but also learning the specific patterns of what a particular reader considers worth highlighting.

Previous approaches to automatic text highlighting have relied on extractive summarization (Nallapati et al., 2017), keyphrase extraction (Meng et al., 2017), or sentence importance scoring (Cheng & Lapata, 2016). However, these methods optimize for generic notions of importance rather than individual reader preferences. Large language models such as Llama 3.3 70B have been applied to this task through fine-tuning, but their computational cost (requiring months of training time) and inference latency make them impractical for large-scale document processing.

We propose a hybrid approach that combines the contextual understanding of a fine-tuned transformer model with the precision of hand-crafted rules and the selectivity of a learned filter. Our system processes academic text in three stages:

1. DeBERTa-v3-large token classification detects candidate highlight spans using BIO (Begin-Inside-Outside) tagging with threshold-based decoding.
2. Rule-based refinement applies keyword patterns, citation detection, and comparison phrase matching to classify highlight types and catch spans the model misses.
3. Gradient Boosting filter selects only those spans that match the target reader's highlighting style, reducing over-prediction by approximately 4x.

2. Related Work

2.1 Token Classification for Span Detection

Token classification with BIO tagging is a well-established approach for named entity recognition (Devlin et al., 2019) and has been adapted for various span detection tasks. DeBERTa (He et al., 2021) introduced disentangled attention mechanisms that improve performance on token-level tasks. DeBERTa-v3-large (He et al., 2023) further improves upon this with replaced token detection pre-training.

2.2 Class Imbalance in Token Classification

Academic highlighting exhibits extreme class imbalance: in our training corpus, 95.6% of tokens belong to the O (no highlight) class, with some highlight types representing less than 0.1% of tokens. Standard approaches include

class-weighted loss functions, focal loss (Lin et al., 2017), and oversampling strategies (Chawla et al., 2002). We employ both focal loss and aggressive oversampling.

2.3 Personalized Text Processing

Learning individual user preferences has been explored in recommendation systems and personalized summarization. Our second-stage filter is related to learning-to-rank approaches (Burges, 2010) where a model learns to predict which candidates a specific user would select.

3. System Architecture

The highlight pipeline consists of five components organized in a sequential processing chain:

```
PDF -> pdf_to_jsonl.py -> JSONL -> highlight_pipeline_v3.py -> Highlighted JSONL -> ApplyHighlights -> Annotated PDF
```

3.1 PDF Text Extraction

The pdf_to_jsonl.py module converts PDF documents to JSONL format using the PyMuPDF library. Text is extracted at the block level, with each text block becoming one JSONL entry containing prompt, completion, and spans fields. The extraction pipeline includes ligature resolution, smart quote and dash normalization, whitespace collapse, and fragment filtering (blocks shorter than 10 characters are discarded).

In our evaluation, this converter processed 6,892 PDF files (2,479,933 paragraphs) in 8.4 minutes at 13.5 files per second, with a 98.4% success rate.

3.2 DeBERTa Token Classifier

3.2.1 Model Architecture

The span detection model is based on DeBERTa-v3-large (304M parameters) with a token classification head producing 17 labels: one O label and B/I tags for each of eight highlight types.

3.2.2 Label Scheme

Type	Description	Training Freq.
normal	General noteworthy text	4.4%
important	Key findings, methods	< 1%
really_important	Critical results, claims	< 0.5%
reference	Citations, references	< 0.3%
confusing	Unclear passages	< 0.1%
pretty_good	Positive assessments	< 0.1%
pretty_bad	Negative assessments	< 0.1%
funny	Humorous content	< 0.1%

3.2.3 Training

The model was trained with AdamW optimizer (learning rate 2e-5), batch size 16, 15 epochs with early stopping (patience 25,000 steps), and focal loss (gamma 2.0) to handle extreme class imbalance. Rare types were oversampled to 3% of the normal count (e.g., pretty_good: 67 to 9,179 samples). Training data comprised 353,626 samples with 39,292 evaluation samples over approximately 132,000 steps.

The focal loss is defined as: FL(p_t) = -alpha_t * (1 - p_t)^gamma * log(p_t), where p_t is the model's estimated probability for the correct class and gamma = 2.0 down-weights easy examples to focus learning on hard cases.

3.2.4 Best Checkpoint Performance

Metric	Value

Binary F1	0.704
Binary Precision	0.595
Binary Recall	0.860
Important F1	0.652
Accuracy	0.806

3.2.5 Inference: Threshold-Based Decoding

Standard argmax decoding proved overly conservative: the model assigned many tokens O-class probabilities in the 45-55% range, causing highlights to be missed. We implemented threshold-based decoding: if $P(O) < 0.98$, select the highest-probability non-O label instead of argmax. Orphan I-tags are recovered as new B-tags. Spans shorter than 5 characters or covering more than 80% of the input text are filtered.

The threshold of 0.98 was determined empirically: even at this aggressive setting, the model maintains 100% recall on test files while reducing predictions from 10x to approximately 2.5x the ground truth count.

3.3 Rule-Based Highlighter

The rule engine applies pattern matching to detect highlights difficult to learn from limited training data. Rules serve two purposes: type refinement (overriding model types with more precise rule-based types) and gap filling (adding matches the model missed).

Rule categories include: Really Important keywords (change, succeed, fail, achieve, insist, maintain, test, determine, etc.), Important keywords (event, when), Comparison phrases (faster than, better than, etc.), Citation patterns (Author (Year), (Author et al., Year), superscript references), Causal patterns (leads to, results in, associated with), and Statistical patterns ($P < 0.05$, 448 ms, $F(1,23) = 4.56$).

Keywords are expanded to clause boundaries with a maximum of 150 characters in each direction to prevent full-paragraph highlighting.

3.4 Span Merging

The pipeline merges model and rule spans: for overlapping spans, rule types override model types. Unmatched rule spans are added. Overlapping spans are deduplicated, keeping the longer or more specifically typed span.

3.5 Second-Stage Gradient Boosting Filter

3.5.1 Motivation

After DeBERTa detection and rule application, the pipeline produces approximately 10x more spans than the target reader would highlight. A second-stage classifier learns the reader's selectivity.

3.5.2 Features

For each candidate span, 19 features are extracted: confidence features (average, min, max, std of non-O probability; average and min O-probability), length features (char length, token count, length ratio), position features (relative start/end), type features (type ID, binary indicators), and text features (digits, parentheses, word count, average word length).

3.5.3 Training

The filter was trained on 80,070 candidate spans (7,219 positive, 72,851 negative) from 1,240 files using Gradient Boosting (500 trees, max depth 6, learning rate 0.1).

3.5.4 Feature Importance

Feature	Importance
min_o (minimum O-probability)	0.199
char_len (character length)	0.187

std_non_o (confidence variation)	0.152
avg_non_o (average confidence)	0.102
avg_o (average O-probability)	0.095
max_non_o	0.071
min_non_o	0.052
token_count	0.042
avg_word_len	0.038
word_count	0.032

3.5.5 Threshold Selection

Threshold	Precision	Recall	Pred/GT Ratio
0.03	0.225	0.948	4.21x
0.10	0.332	0.805	2.43x
0.20	0.432	0.648	1.50x
0.30	0.498	0.532	1.07x
0.50	0.588	0.352	0.60x

4. PDF Annotation Rendering

The ApplyHighlights application (C#/.NET) renders highlighted JSONL output back into annotated PDF files using the O2S PDF4NET library.

4.1 Multi-Line Highlight Support

Highlights spanning multiple PDF lines are rendered as multiple yellow annotation rectangles. The algorithm concatenates all text runs on a page into a single string with tracked character offsets, searches for each highlight text, maps the match back to individual lines, and creates one PDFTextMarkupAnnotation per spanned line. Only the bottom/last annotation box contains the annotation text.

4.2 Two-Column Layout Support

The system detects two-column layouts by comparing line X positions against the page midpoint. Left column highlights receive margin images on the left side; right column highlights receive margin images on the right side. Overlap detection prevents images from stacking.

4.3 Type Annotation Images and Text

Each highlight type is associated with a margin annotation image (CheckMark for important, DoubleCheckMark for really important, ReferenceDot for reference, QuestionMark for confusing, ExclamationPoint for pretty good, Frown for pretty bad, SmileyFace for funny). Vertical images are dynamically scaled to match the highlight span height. Annotation contents follow the format: [*Type*] highlighted text.

5. Evaluation

5.1 Conflict Adaptation Test File

On the initial test file (47 lines, 17 ground truth spans), the complete pipeline with filter threshold 0.10 achieved:

Metric	No Filter	With Filter
Predictions	88	42
True Positives	17/17	17/17
Precision	0.193	0.405
Recall	1.000	1.000

F1	0.324	0.576
Pred/GT Ratio	5.18x	2.47x

5.2 Full Corpus Evaluation

Across 11 annotated documents (13,940 lines, 1,290 ground truth spans):

File	GT	Pred	Recall	F1	Ratio
conditional_memory	123	259	0.902	0.581	2.1x
unicog	72	152	0.889	0.571	2.1x
policy_of_thoughts	88	214	0.909	0.530	2.4x
evolving_cognitive	158	347	0.823	0.515	2.2x
affect_dynamics	494	3685	0.931	0.220	7.5x
TOTAL	1290	6297	0.898	0.305	4.9x

5.3 Processing Speed

Component	Speed
PDF to JSONL conversion	13.5 files/sec
DeBERTa + rules + filter	~35 lines/sec
Complete paper (avg 366 lines)	~10 seconds

5.4 Scale Validation

The pipeline has been deployed on 6,779 academic papers (2,479,933 paragraphs) and 341 books (918,834 paragraphs).

6. Discussion

6.1 Strengths

The hybrid approach leverages complementary strengths of neural and rule-based methods. DeBERTa provides contextual understanding of noteworthy text, while rules precisely capture known reader preferences. The second-stage filter learns selectivity from large-scale annotation data. The threshold-based decoding strategy was critical: standard argmax decoding achieved only 1-7% recall in initial testing, while threshold decoding at 0.98 recovered 100% recall.

6.2 Limitations

1. Precision gap: The system predicts 2-7x more spans than the target reader. The remaining false positives are generally informative sentences the model considers noteworthy but the reader chose not to highlight.

2. Span boundary mismatch: The reader highlights specific short phrases; the model often highlights broader clauses containing those phrases.

3. Type classification depends heavily on rules rather than the model for rare classes.

4. Full-book performance: The prediction ratio increases for longer documents (7.5x for 8,116-line book vs. 2.1x for shorter papers).

6.3 Comparison to Previous Approach

Aspect	Llama 70B	DeBERTa Pipeline
Model size	70B (4-bit)	304M
Training time	~2 months	~3 days
GPU memory	~24 GB	~5 GB
Inference speed	~1 line/sec	~35 lines/sec
Recall	Not evaluated	89.8%

7. Conclusion

We have presented a practical system for personalized academic highlight detection that combines the contextual understanding of a fine-tuned DeBERTa model with hand-crafted rules and a learned selectivity filter. The system achieves near-perfect recall (89.8-100%) on the target reader's highlights while maintaining manageable false positive rates. The complete pipeline, including PDF conversion and annotation rendering, has been validated on a corpus of over 7,000 academic documents.

The key insight is that personalized highlighting is best approached as a two-stage problem: first detecting all potentially noteworthy spans (high recall), then filtering to match individual preferences (improved precision). The rule engine provides an interpretable and extensible middle layer that captures preferences difficult to learn from data alone.

References

Burges, C. J. C. (2010). From RankNet to LambdaRank to LambdaMART: An Overview. Microsoft Research Technical Report.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. JAIR, 16, 321-357.

Cheng, J., & Lapata, M. (2016). Neural Summarization by Extracting Sentences and Words. ACL.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers. NAACL.

He, P., Liu, X., Gao, J., & Chen, W. (2021). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. ICLR.

He, P., Gao, J., & Chen, W. (2023). DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training. ICLR.

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollar, P. (2017). Focal Loss for Dense Object Detection. ICCV.

Meng, R., Zhao, S., Han, S., et al. (2017). Deep Keyphrase Generation. ACL.

Nallapati, R., Zhai, F., & Zhou, B. (2017). SummaRuNNer: A Recurrent Neural Network based Sequence Model. AAAI.

Appendix A: Software Components

Component	Language	Lines	Description
highlight_pipeline_v3.py	Python	733	Core hybrid pipeline
train_deberta_v3.py	Python	653	DeBERTa training
pdf_to_jsonl.py	Python	175	PDF text extraction
test_pipeline_on_file.py	Python	116	Evaluation harness
ApplyHighlights/Program.cs	C#	491	PDF annotation renderer
span_filter_model.pkl	-	2.4 MB	Trained GB filter

Appendix B: Highlight Type Image Annotations

The PDF annotation system uses visual margin indicators for each highlight type:

Image	Size	Type
CheckMark	24x24	Important
DoubleCheckMark	24x24	Really Important
ReferenceDot	5x5	Reference
QuestionMark	7x24 (scaled)	Confusing
ExclamationPoint	5x24 (scaled)	Pretty Good
Frown	24x24	Pretty Bad
SmileyFace	24x24	Funny
VerticalBar	5x24 (scaled)	Generic marker